Refining an Almost Clean Translation Memory Helps Machine Translation

Shivendra Bhardwaj shivendra.bhardwaj@umontreal.ca
David Alfonso-Hermelo david.alfonso.hermelo@gmail.com
Philippe Langlais felipe@iro.umontreal.ca
RALI/DIRO, Université de Montréal, Montréal, QC H3C 3J7, Canada

Gabriel Bernier-Colborne Gabriel.Bernier-Colborne@nrc-cnrc.gc.ca
Cyril Goutte Cyril.Goutte@nrc-cnrc.gc.ca
Michel Simard Michel.Simard@nrc-cnrc.gc.ca
Multilingual Text Processing, National Research Council, Ottawa, ON K1A 0R6, Canada

Abstract

While recent studies have been dedicated to cleaning very noisy parallel corpora to improve Machine Translation training, in this work we focus on filtering a large and mostly clean Translation Memory. This problem of practical interest has not received much consideration from the community, in contrast with, for example, filtering large web-mined parallel corpora. We experiment with an extensive, multi-domain proprietary Translation Memory and compare five approaches involving deep-, feature-, and heuristic-based solutions. We propose two ways of evaluating this task, manual annotation and resulting Machine Translation quality. We report significant gains over a state-of-the-art, off-the-shelf cleaning system, using two MT engines.

1 Introduction

Major language service providers that handle huge numbers of translation requests in various domains typically call upon a large pool of translators (internal and freelance) whose translations are fed into one or several internal *Translation Memories* (TM). Translators access these TMs through a dedicated interface to match requested translations with previously completed ones, speeding up the translation process. Given their provenance and purpose, it is normally assumed that TMs are mostly exempt of incorrect translations (henceforth "*noise*"). Unfortunately, because of the high number of translators involved, with varying levels of expertise, tight deadlines, and other technological issues, noise inevitably accumulates over the years.

TM noise falls under two broad categories. First, *mechanical noise*, which occurs because of the pipeline used to populate the TM, whereby text is extracted from source and target documents, segmented into sentences and then aligned — all three processes producing occasional errors. Second, *human-induced noise*, which can arise for a variety of reasons, including spelling or morphosyntax that do not meet established norms, missing translation units on the target side, as well as typical translation errors such as calques, the use of false friends, etc. These errors reduce the usefulness of a TM, and motivate our investigation into whether *Parallel Corpus Filtering* methods may be useful to increase the quality of a large TM. This contrasts with the typical use of these methods, which are more commonly applied to large, very noisy bilingual corpora automatically extracted from the Web (or other uncontrolled sources).

In the following section (Sec. 2), we discuss related work in more details. We then describe

the large TM on which our experiments are based in Section 3, and the corpus filtering methods we implemented in Section 4. We evaluate the performance of these methods by measuring their impact on Neural Machine Translation (NMT) in Section 5, and report our results and analysis in Section 6. We show that, surprisingly, significant translation quality gains can be obtained by cleaning an "already clean" Translation Memory.

2 Related Work

2.1 Identifying Translation

Munteanu and Marcu (2005) presented an early successful method to identify translated sentence pairs (SPs) in a comparable corpus, using a feature-based classifier trained in a supervised way. Features included source-to-target length ratio, bilingual lexicon matches, and a set of features based on IBM word translation models (Brown et al., 1993). The authors showed that the parallel material mined from news extracted over the web improved a downstream statistical translation engine. Progress in deep learning methods recently led to a number of classifiers trained without feature engineering. Notably, Grégoire and Langlais (2018) describe a siamese recurrent neural network that encodes source and target sentences into vectors that are then fed through a non-linear transformation in order to classify a sentence pair as parallel or not. The authors show that training such a model yields better performance than the aforementioned approach, and that adding parallel material extracted from Wikipedia using this model leads to systematic (although modest) gains in both statistical and neural machine translation performance. While these studies convincingly show that parallel sentences can be mined from comparable corpora, it remains unclear whether these methods are also useful for filtering out noise from a relatively clean translation memory.

2.2 Filtering Out Noise

In an early attempt to tackle this issue, Macklovitch (1994) used simple heuristics to detect specific problems observed in real (professional) translations, such as errors in numerical entities, calques, or abnormal translation sizes. Barbu (2015) extended this line of work, proposing 17 features, some based on formal clues (e.g., presence/absence of XML tags, emails, URLs, numbers, capital letters or punctuation), or using external resources (Bing translation API and the language detector Cybozu). Using these features, classifiers were trained to recognize bad translations, using a very small training set (1243 sentence pairs). The best model achieved 81% F-score on 309 test sentence pairs. The author concluded that applying it on MyMemory (Trombetti, 2009) would filter out too many good sentence pairs.

Jalili Sabet et al. (2016) introduced a fully unsupervised Translation Memory cleaning tool called TMOP. It uses 25 different features, some adapted from Barbu (2015), others based on the work of de Souza et al. (2014) and focus on estimating the quality of the translation. They also use multilingual word embeddings, following Søgaard et al. (2015). Each feature acts as a filter for which a score is returned, then TMOP transforms these scores into a final decision (Section 4.3 provides more details). Tested on a subset of the English-Italian MyMemory, their system produced results comparable to Barbu (2015) while being unsupervised.

Recently, there has been increased interest in filtering very noisy, usually web-mined parallel corpora using unsupervised deep learning. Chaudhary et al. (2019) trained multilingual sentence embeddings using LASER (Artetxe and Schwenk, 2018), scoring sentence pairs using an ensemble of evaluation methods like Zipporah (Xu and Koehn, 2017), Bicleaner (Sánchez-Cartagena et al., 2018), and dual conditional cross-entropy filtering (Junczys-Dowmunt, 2018). We tried this approach in our work, but found LASER alone to be significantly more efficient: In the evaluation reported in Table 2 (Section 6), the ensemble approach reached an accuracy of 0.54 vs. 0.84 for LASER alone. Wang et al. (2018) proposed another state-of-the-art online data

Corpus	TM-XL	META-H	BALANCED	MT-TRAIN	MT-TEST	TEST2021
#SPs	139.5M	18.9M	7M	14M	10k	2021
#Types (fr)	1.1M	571k	463k	388k	10.7k	3.48k
#Types (en)	1.4M	681k	444k	466k	12.9k	4.30k

Table 1: Corpora statistics. "#SPs" is the number of sentence pairs. "#Types" are the number of space-separated strings of 15 characters or less containing only alphabetical symbols.

`ori`	If you feel like sleeping , stand up and move to back .
`cor`	If you feel**s** **just napp**ing, **S**tand up and move**s** to **ab**ck .
`ori`	The government of Canada will match your contribution dollar for dollar .
`cor`	The govern**n**ment of **Quebec** will match your **C**ontribution doll**l**ar for dollar.

Figure 1: Examples of original (`ori`) and corrupted (`cor`) sentences.

selection method for de-noising training material and adapting to a specific domain, but this is less suited for large TMs with many domains (200 in our corpus, see below).

3 Datasets

For our experiments, we obtained access to a large English-French corporate TM from a major language service provider (LSP). From this, we extracted various datasets (see also Tab. 1):

TM-XL: From over 1.8M TMX files across more than 200 broad domains (e.g. health, environment), we extracted a total 139 454 913 sentence pairs (SPs). In the TM system used by the LSP, translators may flag a problem with a sentence pair, which marks the entire TMX file containing it as problematic. Flagged material represents 7.7% of all SPs, but we don't know which SP from a flagged TMX was the cause of the problem.

MT-TRAIN: For training the translation engines used for evaluation (Sec. 5), we sampled 14M sentence pairs from TM-XL using stratified sampling to get comparable amounts of SPs in each domain. Of these, 4.3M sentence pairs were sampled from the flagged part of the corpus, so that we could monitor if this material impacts translation.

MT-TEST: We also sampled a test set of 10 000 sentence pairs from TM-XL. To minimize the noise in the test set, our sampling excluded sentences from flagged files as well as SPs labeled as noise by heuristics (see Section 4.1).

META-H: To obtain a training set for our supervised classifiers, we applied the two meta-heuristics described in Sec. 4.1 on TM-XL and identified 17.7M sentence pairs labelled as good and 1.2M labelled as bad, for a total corpus of 18.9M annotated SPs.

BALANCED: Due to the heuristics deployed, most bad SPs in META-H feature obvious errors (gibberish, typos, non-translations, etc.), but lack simple misalignment or subtle translation errors. We extend META-H automatically by: a) 1.15M English sentences paired with a random French sentence (misalignments), and b) 1.15M SPs where each token of 4 or more characters is replaced with one of its top five nearest neighbours in a space of `fastText` word embeddings (Bojanowski et al., 2016) (see corrupted SPs in Fig. 1). Joining those 2.3M artificially generated pairs, the 1.2M bad pairs from META-H, and 3.5M SPs sampled from the 17.7M good pairs in META-H, yields a BALANCED corpus of 7M SPs.

en	Section 34 verification and certification
fr	Fiches de spécimen de signature.
en	Native Women's Association of Canada.
fr	Anaya, Doc. NU A/HRC/9/9, 11 août 2008. Native Women's Association of Canada.
en	Since 2005, we have received some $1.4 billion to purchase 17 vessels.
fr	Conflicting sovereignty claims to the Arctic are resulting in a race to the North.
en	`OPnj#ϕ'L O- nSk`
fr	`i@n [u05ce \x9b}`

Figure 2: Example noise in TM-XL. Top to bottom: poorly aligned section heads; segmentation problem leading to partial alignment; complete misalignment; character encoding issue.

TEST2021: We conducted targeted manual evaluations on a reduced but representative sample of 2021 SPs, used for evaluation purposes (cf. Sec. 4.1). TEST2021 contains 1182 (58.5%) good and 839 (41.5%) bad SPs.

4 Noisy Sentence Pair Detection

We compare five approaches to identify and filter out noisy sentence pairs.

4.1 Heuristics

A visual inspection[1] of TM-XL led us to identify specific problems that could be detected by rules (see Fig. 2 for examples). We developed 13 heuristics, most of which similar to those implemented in other systems described in Section 2. All heuristics take as input a sentence pair (SP) and produce a score between 0 (noisy SP) and 1 (good SP).

Some heuristics reward matching numerical expressions (NUM), cognates (COG), punctuation (PUNC), or URLs (URL) across pairs of sentences. Two heuristics exploit lists of matching tokens between French and English: one for stop words, based on a lexicon of 93 entries (e.g. *the* / *la*, *le*, *les*) (STOP), one for general vocabulary (LEX) based on a 60k-word bilingual lexicon. One heuristic (ION) matches words ending with suffix *-ion* in French with an identical suffix in English (e.g., *félicitations* / *congratulations*). Other heuristics aim at detecting common problems: LEN checks the source-to-target length ratio (in words); GIBB detects character encoding problems (last example in Fig. 2); MONO flags pairs where the target segment contains source-language words, suggesting untranslated material; FRIEND penalizes SPs containing false friends, based on a lexicon of 175 entries (e.g. *fabric*/*fabrique*). We also noticed that segmentation issues within tables of contents led to alignment errors: heuristic TOC aims at filtering these out. Finally, a rudimentary proxy to spell checking (SPELL) counts the number of correctly spelled tokens, using a list of words seen at least 1000 times in Wikipedia.

We evaluate the performance of each heuristic using a set of 1721 sentences pairs: 1321 randomly sampled from TM-XL, plus 400 picked for specific problems. We annotated these 1721 sentence pairs and used them to adjust the thresholds of our heuristics and to select the optimal combination of heuristics to discriminate good SPs from bad. For detecting good SPs, the solution that worked best is a weighted combination of 4 heuristics: NUM, LEX, ION, and PUNC, while the meta-heuristic that worked best to predict problematic SPs is a mix of 9 heuristics. To further evaluate these two "meta-heuristics", we manually inspected a random sample of 150 SPs selected by each (i.e. identified as good or bad respectively) and found 115 good SPs in the former sample (76.7%), and 121 bad SPs in the latter (80.7%).

[1] Separate from the Manual evaluation.

4.2 Feature-based Classifiers

We trained support vector machines (SVM) and random forest classifiers on the BALANCED corpus to detect noisy pairs of segments. The features for each SP are the scores (between 0 and 1) of each of the 13 heuristics in Sec. 4.1, plus some intermediate values produced while computing the heuristics, for a total of 60 features. We added two features: the percentage of heuristics that label the pair as *good* (resp. *bad*). We trained the classifiers with `scikit-learn`[2] on standard desktop CPUs, which took approximately 10 hours per classifier.

4.3 TMOP

TMOP (Jalili Sabet et al., 2016) uses 25 binary functions meant to capture misalignments, poor translation quality or large semantic distance between source and target. Three ready-made configurations control how these functions combine into a final decision. The configuration we use classifies an SP as noise if at least five functions signal a problem. In our experiments, this configuration was by far the most useful: the "one reject" configuration produced far too many false negatives, while the "majority vote" hardly detected any *bad* sentence pair. Similarly to Munteanu and Marcu (2005), TMOP relies (among other things) on IBM Model features computed with MGIZA.[3] Running MGIZA on the 14M SPs in MT-TRAIN on a 16-core cluster equipped with 70Gb of memory took 13 days. After alignment, TMOP ran on a dedicated cluster of 32 CPUs with 300Gb of RAM, and took 5 more days, including 6 hours to compute embeddings. Therefore, applying TMOP on the full TM-XL corpus would be rather challenging.

4.4 Deep-Learning Classifier

We reimplemented the model of Grégoire and Langlais (2018) in Keras (Chollet et al., 2015), introducing a few variants we found useful. The model architecture consists of two bidirectional LSTMs (Hochreiter and Schmidhuber, 1997), each with 300 hidden units encoding sentences into two continuous vector representations. In their original paper, Grégoire and Langlais (2018) use 512-dimensional word embeddings and 512-dimensional recurrent states and learn the word embeddings from scratch. For easier and faster training, we adapt pre-trained 300-dimensional `fastText` embbedings. Also, we do not tie the parameters of the two encoders, contrary to Grégoire and Langlais (2018). Source and target representations are then fed into a Feed-Forward Neural Network with two hidden layers of 150 and 75 units (respectively), followed by a sigmoid activation function, which outputs the probability that the input SP is well aligned. We trained the model using the Adadelta optimizer (Zeiler, 2012) with gradient clipping (clipped at 5) to avoid exploding gradient and a batch of size 300, which took about 2.5 hours using 4 Tesla V100-SXM2 for 10 epochs.

4.5 LASER

We also use the LASER toolkit[4] (Artetxe and Schwenk, 2018) to detect noisy pairs. LASER is a bi-LSTM encoder trained on data from 93 different languages, written in 23 alphabets, such that semantically similar sentences in different languages are close in the embedding space. For each source-language sentence s_i in MT-TRAIN, we use the multilingual-similarity search (MSS) method from the toolkit to find the closest target-language sentence t_j in the embedding space. If $i = j$, the sentence pair is considered *good*, otherwise it is *bad* and filtered out. Obviously, we could investigate a less stringent scenario, which we leave for future work. This is entirely unsupervised, using the model *as is*, out of the box. Running this method on one

[2] `https://scikit-learn.org/stable/`.

[3] `http://www.cs.cmu.edu/~qing/giza/`.

[4] `https://github.com/facebookresearch/LASER`.

Tesla V100-SXM2 took approximately 14 hours, despite the quadratic number of comparisons involved.

5 Neural Machine Translation Models

We evaluate the quality of a TM using the performance of a translation engine trained on it as a task-based proxy. To reach conclusions that are independent of a specific system, we experiment with two very different neural translation models: XLM, a deep transformer model, and ConvS2S, a convolutional seq2seq model. Typical training time on 4 Tesla V100-SXM2 was 22-30 hours for XLM and 72-96 hours for ConvS2S, depending on the dataset.

5.1 Cross-lingual Language Model

Lample and Conneau (2019) proposed a supervised model, the Translation Language Modeling (TLM), tackling cross-lingual pre-training in a way similar to BERT (Devlin et al., 2018), with notable differences. First, XLM is based on a shared source-target sub-word vocabulary, computed using byte pair encoding (BPE) (Sennrich et al., 2016). We used the 60k BPE vocabulary from the pre-trained language model.[5] Second, XLM is trained to predict both source and target masked words, leveraging surrounding words and context on both sides and encouraging the model to align source and target representations. Third, XLM embeds the tokens *and* their position, building a relationship between the related tokens in both languages. XLM is implemented in PyTorch and supports distributed training on multiple GPUs.[6] We modified the original pre-processing code so that XLM accepts a parallel corpus for training TLM. The translation is produced using a beam search of width 6 and unity length penalty.

5.2 Convolutional Sequence to Sequence

The predominant method for sequence to sequence (seq2seq) learning is to map an input sequence to an output sequence of variable length via a recurrent neural network such as an LSTM (Hochreiter and Schmidhuber, 1997). Gehring et al. (2017) showed that convolutional neural networks (CNN) could also be used for seq2seq. The ConvS2S model uses CNNs with Gated Linear Units (Dauphin et al., 2016) for both the encoder and decoder, and includes a multi-step attention layer. We used the fairseq toolkit (Ott et al., 2019), with a source and target vocabulary of 60k BPE types. The translation is generated by a beam-search decoder with log-likelihood scores normalized by sentence length.

6 Experimental Results

We now report results on noisy SP detection and its impact on machine translation.

6.1 Sentence Pair Detection

Table 2 reports the tested methods' accuracy on the manually annotated TEST2021. The metaheuristics alone perform worst. Training RF and SVM classifiers on top of heuristics features clearly helps, with the SVM showing a slight advantage. TMOP delivers comparable results, suggesting that combining heuristics, word-based translation and embeddings makes a good detector. The bi-LSTM model, without any feature engineering, yields much better results overall, confirming observations by Grégoire and Langlais (2018) on artificial data. Surprisingly, the best results are obtained by the fully unsupervised LASER. Of course, TEST2021 is a rather small test set, and results here should be taken with a grain of salt.

[5] Training TLM without pre-training proved unstable. Back translation gave better results, at a high training cost.

[6] https://github.com/facebookresearch/XLM.git.

Method	TMOP*	meta-h*	RF	SVM	bi-LSTM	LASER*
Accuracy	0.60	0.42	0.60	0.63	0.79	0.84

Table 2: Accuracy on TEST2021. See text for method details (* are unsupervised). 95% error bars on estimates are $\approx \pm 2\%$.

Train set	MT-TRAIN	−flagged	TMOP	meta-h	SVM	bi-LSTM	LASER	∩ALL
#SP (M)	14	9.67	13.38	8.15	7.50	6.13	9.65	5.80
XLM	36.25	36.29	36.49	36.80	36.53	‡37.52	‡37.23	‡37.57
ConvS2S	33.04	33.33	33.51	†33.78	†33.91	‡33.96	33.58	†33.93

Table 3: BLEU scores of the XLM and ConvS2S translation engines. −flagged is MT-TRAIN without the flagged documents (Sec. 3). ‡ (resp. †) means improvement over MT-TRAIN is significant at the 99% (resp. 95%) confidence level using `multeval` (Clark et al., 2011).

6.2 Machine Translation Evaluation

Machine translation is used as an extrinsic evaluation of our corpus cleaning methods. Note that the goal is not to optimize the use of a TM on a single MT engine as in (Cao and Xiong, 2018, for example), but to evaluate systematic differences in MT performance before and after cleaning, using two very different state-of-the-art neural translation engines. Similarly, we acknowledge that neither of the MT systems used here is optimized to reach current state-of-the-art (although in separate experiments, we observed that XLM came close). We are mainly interested in observing relative performance differences for different filtering, and as we will see below, these trends are consistent, despite significant differences in absolute performance.

Table 3 shows BLEU scores obtained by XLM and ConvS2S on portions of MT-TRAIN produced by different filtering approaches. ConvS2S is consistently about 3 BLEU points worse than XLM, but reflects the same trends overall. Removing the flagged material (-flagged) from MT-TRAIN hardly impacts BLEU, which supports our observations that the flagged material was generally of good quality. Out of the box, TMOP filters out very few SPs (less than 5%), without producing any significant gain in BLEU. Some adaptation to the Translation Memory may be needed to deploy it efficiently. Table 3 also shows that all methods described in Section 4 filter a significant portion of MT-TRAIN (31% to 56%), resulting in BLEU gains that are sometimes highly significant. This is already a surprising outcome, for a corpus sampled from a TM, which is supposed to be already clean. The largest gains come from the supervised bi-LSTM approach, which also filters out more material, with the unsupervised LASER not far behind. Using the intersection of all our filters (last column in Tab. 3) filters a few more SPs, and further improves performance. The final BLEU gain is +1.22 for XLM (+0.89 for ConvS2S) with only 42% of MT-TRAIN remaining. It is interesting to note that -flagged and LASER yield very similar amounts of training material, but the latter results in significantly higher BLEU. This suggests that 1) as mentioned previously, the flagged material contains clean material that is useful to the MT engine, while 2) the non-flagged portion of the TM contains noisy material that has not been flagged by translators, but can be filtered out to improve MT performance.

Fig. 3 plots the BLEU scores on the test set (versus epochs) for XLM translation engines trained on the different portions of MT-TRAIN. We observe similar training curves, including a sharp increase in performance at epoch 10, for all systems, and systematic gains at each epoch.

6.3 Analysis

We now investigate various aspects of the cleaning process.

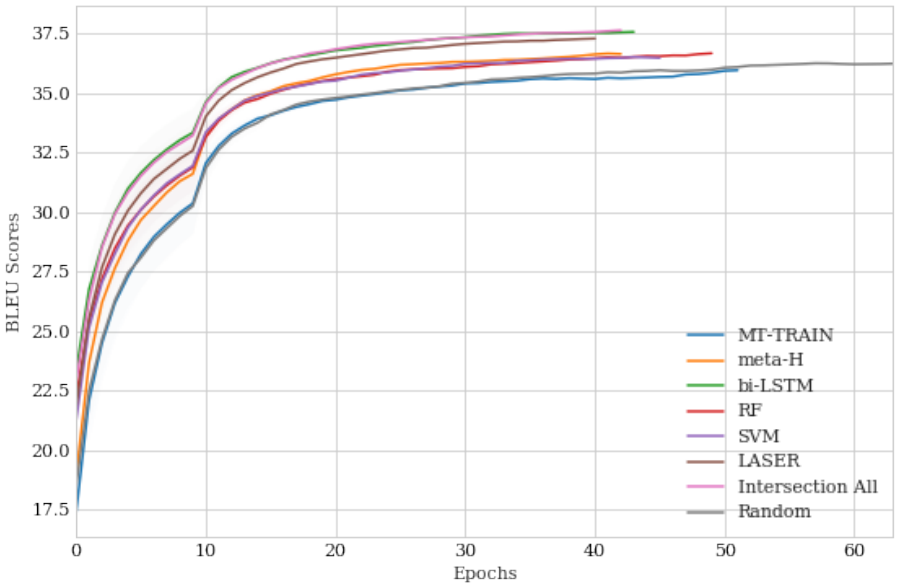

Figure 3: BLEU scores of XLM versus epochs for the different training sets we considered.

Domain Specificity. One possible concern is that the filtering is essentially performing some kind of adaptation to the test set. We first note that MT-TRAIN and MT-TEST are both sampled from TM-XL and are similarly balanced according to domain information recorded in the TM. We also compare the domain distribution in the original TM to that in MT-TRAIN and its various filtered versions, using the Kullback-Leibler (KL) divergence: All filtered subsets yield a divergence close to that computed between MT-TRAIN and the original TM (KL $= 0.0110$). This suggest that the domain distribution is not affected by any filtering method.

Combining Methods. Different filtering methods remove different amounts and portions of MT-TRAIN (Tab. 3). We checked agreement between the meta-H, bi-LSTM and LASER filters and observe that 93.4% of SPs identified as noise by LASER were also labeled as such by bi-LSTM, while the agreement on noise between LASER and either meta-H or SVM is approximately 65%. Removing SPs labeled as noise by any of meta-H, bi-LSTM or LASER (named $\cap$ALL in Tab. 3) results in a corpus of 5.8M sentence pairs, which yields BLEU scores similar to bi-LSTM. This suggests that the deep learning methods do not benefit from the other techniques, although they allow to further clean the corpus with no performance loss. As a sanity check, we randomly select a subset of 5.8M SPs from MT-TRAIN, and observe a drop in translation performance of -1.26 and -0.93 BLEU for XLM and ConvS2S respectively (not shown in Table 3). This further supports the claim that our methods do perform a useful cleaning of the translation memory.

Unknown Words. Both translation engines use a fixed set of 60k BPE subword units. Some target words in the training material can not be reproduced using that limited vocabulary. For the XLM translation engine on MT-TRAIN, we count 76k such token types. Manual inspection shows that the vast majority are gibberish words, e.g. *t0DÉlmsäoyCœ*, likely document conversion problems. Filtering the training material with SVM, bi-LSTM and LASER, reduces the number of unrepresentable token types to around 3k, 450 and 17k, respectively. This indicates that our cleaning approaches (especially bi-LSTM) efficiently reduce the number of unknown,

gibberish words.

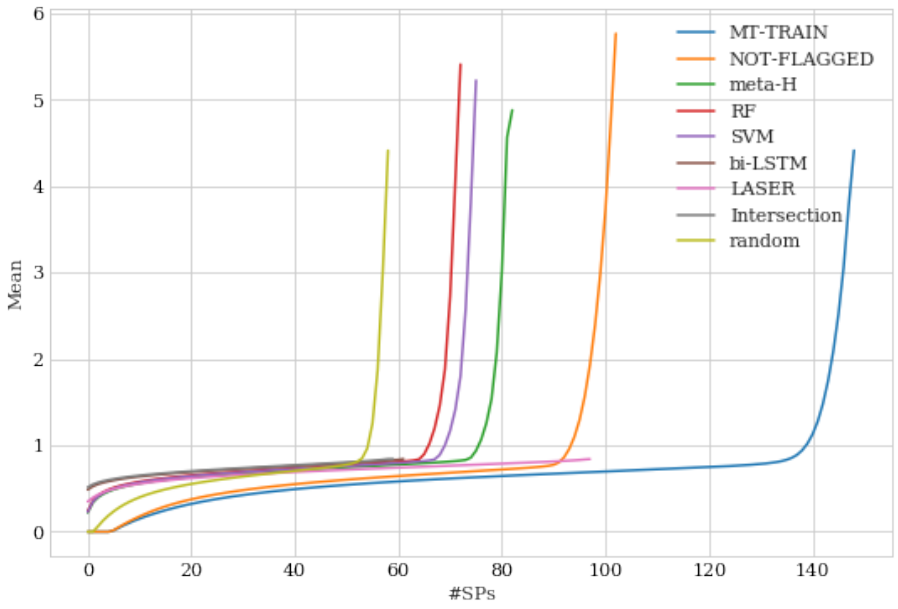

Figure 4: Average length-ratio of sentence pairs of the different sub-corpora of MT-TRAIN as a function of the number of slices of 100k SPs considered.

Length-ratio. Figure 4 shows the length ratio ($|en|/|fr|$) of sentence pairs in portions of MT-TRAIN filtered in different ways, in blocks of 100k SPs sorted by increasing length ratio. The curve for MT-TRAIN starts near zero (much longer French sentence) flattens slightly below 1 as expected, and peeks around 4 (longer English sentences). Both extremes very likely indicate alignment problems. Applying meta-heuristics or feature-based classifiers reduces near-zero and high ratios to some extent. Noticeably, bi-LSTM and LASER remove most of the SPs with extreme length ratios, leading to an average around 0.8 (French is about 20% longer than English, on average). This suggests that cleaning is indeed being performed, removing extreme misalignments.

Other Evidence of Cleaning. MT-TRAIN contains 25 203 sentence pairs with a *www* token in English, but not on the French side, versus 57 925 with matching *www* token, a ratio of 43.5%. SVM, bi-LSTM and LASER lower that ratio to 10.5%, 3.1% and 4.9% respectively, after filtering. We also manually annotated 100 sentence pairs from $\cap$ALL, i.e. classified as good by all approaches, and 100 sentence pairs classified as noise by bi-LSTM and LASER. We found 16 false positives in the former and 33 false negatives in the latter. This suggests that deep learning methods are excessively strict noise detectors. This, however, does not seem to impact MT performance adversely.

Reproducibility. While we had the opportunity to work on a large, high quality professional TM, we realize that our results can not be replicated exactly. By nature, large professional TMs are proprietary and not easily shared. We argue however that one can easily *reproduce* (Drummond, 2009) our experiments on another corpus or TM, using the information from Sections 4 to 5. In addition, the main building blocks for the better-performing filtering and translation pipelines are publicly available.

Generalizability to other languages. For the same reason (access to a large professional TM), we only performed experiments on a single language pair comprising similar languages with, e.g., many cognates. We acknowledge that it is an interesting and important issue to establish whether a similar approach applies to languages with different characteristics, e.g. complex morphology, or differing word order or sentence structure. The availability of deep learning models such as LASER on a (relatively) large number of languages should make it straightforward to experiment on many other language pairs to check whether the results presented in this paper generalize.

7 Conclusions

We explored several ways to filter a mostly clean Translation Memory, used daily by professional translators. We showed that (pre-trained) LASER and (trained in-house) bi-LSTM are able to discriminate noisy sentence pairs from clean ones with high accuracy. The former is unsupervised, delivering the best results on our small scale manual evaluation. Both methods outperform heuristics devised specifically for this task, feature-based classifiers trained with supervision, as well as the TMOP system which turned out to be very challenging to deploy. We also showed that filtering the noisy SPs results in machine translation gains. Deep learning methods allow significant gains in BLEU: the bi-LSTM classifier filters out over half the training material, and yields a gain of over one BLEU point. Future work includes better characterizing, modeling and generating subtle noise in misaligned segments, in order to build better detectors.

Acknowledgements

We wish to thank the anonymous reviewers for their insightful and helpful comments.

References

Artetxe, M. and Schwenk, H. (2018). Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *arXiv:1812.10464*.

Barbu, E. (2015). Spotting false translation segments in translation memories. In *Proc. Workshop on Natural Language Processing for Translation Memories*, pages 9–16.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv:1607.04606*.

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Comput. Ling.*, 19(2):263–311.

Cao, Q. and Xiong, D. (2018). Encoding gated translation memory into neural machine translation. In *Proc. Empirical Methods in Natural Language Processing*, pages 3042–3047.

Chaudhary, V., Tang, Y., Guzmán, F., Schwenk, H., and Koehn, P. (2019). Low-resource corpus filtering using multilingual sentence embeddings. *arXiv:1906.08885*.

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Clark, J. H., Dyer, C., Lavie, A., and Smith, N. A. (2011). Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proc. 49th Annual Meeting of the ACL: Human Language Technologies*, pages 176–181, Portland, OR.

Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. (2016). Language modeling with gated convolutional networks. *arXiv:1612.08083*.

de Souza, J. G., Turchi, M., and Negri, M. (2014). Machine translation quality estimation across domains. In *Proc. 25th Intl. Conf. on Computational Linguistics (COLING)*, pages 409–420.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*.

Drummond, C. (2009). Replicability is not reproducibility: nor is it good science. In *Proceedings of the Evaluation Methods for Machine Learning Workshop at the 26th ICML*.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. *arXiv:1705.03122*.

Grégoire, F. and Langlais, P. (2018). Extracting parallel sentences with bidirectional recurrent neural networks to improve machine translation. In *Proc. 27th Intl Conf. on Computational Linguistics (COLING)*, pages 1442–1453, Santa Fe, NM.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Jalili Sabet, M., Negri, M., Turchi, M., de Souza, J. G., and Federico, M. (2016). TMop: a tool for unsupervised translation memory cleaning. In *Proc. of the ACL, System Demonstrations*.

Junczys-Dowmunt, M. (2018). Dual conditional cross-entropy filtering of noisy parallel corpora. In *Proc. 3rd Conf. on Machine Translation*, pages 888–895, Belgium, Brussels.

Lample, G. and Conneau, A. (2019). Cross-lingual Language Model Pretraining. *arXiv:1901.07291*.

Macklovitch, E. (1994). Using bi-textual alignment for translation validation: the TransCheck system. In *1st Conf. Association for Machine Translation in the Americas*, Columbia, USA.

Munteanu, D. S. and Marcu, D. (2005). Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4):477–504.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. (2019). fairseq: A fast, extensible toolkit for sequence modeling. In *Proc. 2019 Conf. North American Chapter of the ACL (Demonstrations)*, pages 48–53, Minneapolis, MN.

Sánchez-Cartagena, V. M., Bañón, M., Ortiz-Rojas, S., and Ramírez, G. (2018). Prompsit's submission to WMT 2018 parallel corpus filtering shared task. In *Proc. 3rd Conf. on Machine Translation*, pages 955–962, Belgium, Brussels.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proc. 54th Annual Meeting of the ACL*, pages 1715–1725.

Søgaard, A., Agić, Ž., Alonso, H. M., Plank, B., Bohnet, B., and Johannsen, A. (2015). Inverted indexing for cross-lingual NLP. In *Proc. 53rd Meeting of the ACL (ACL-IJCNLP)*.

Trombetti, M. (2009). Creating the world's largest translation memory. In *MT Summit XII*.

Wang, W., Watanabe, T., Hughes, M., Nakagawa, T., and Chelba, C. (2018). Denoising neural machine translation training with trusted data and online data selection. In *Proc. 3rd Conf. on Machine Translation*, pages 133–143, Brussels, Belgium.

Xu, H. and Koehn, P. (2017). Zipporah: a fast and scalable data cleaning system for noisy web-crawled parallel corpora. In *Proc. Empirical Methods in Natural Language Processing*, pages 2945–2950, Copenhagen, Denmark.

Zeiler, M. D. (2012). Adadelta: An adaptive learning rate method. *arXiv:1212.5701*.